ChatGPT

Abstract

TGC v2 is a **machine-layer deterministic trust governance protocol** that formalizes and automates decentralized decision-making with an emphasis on structural trust. In TGC v2, all governance actions and trust relationships are recorded on an immutable ledger and are transparently observable by every participant. The protocol replaces ad-hoc or probabilistic trust assumptions with explicitly encoded **structural trust**, meaning that the system's rules and cryptographic design – rather than personal discretion – guarantee the integrity of decisions. We describe the TGC v2 system model, define its governance rules and processes in detail, and provide mathematical formulations for key mechanisms such as voting thresholds and trust delegation transitions. We prove theoretical properties of the protocol including determinism, transparency, and security, and we illustrate how "**trust is structure**" in TGC v2: trust emerges from verifiable algorithmic processes (e.g. multi-signature quorums and delegated voting graphs) rather than subjective judgment [1] [2]. Finally, we explore use cases ranging from decentralized autonomous organizations (DAOs) to AI agent coordination, and discuss limitations of the approach. TGC v2 demonstrates that robust governance can be achieved through machine-layer rules that enforce collective trust, offering a foundation for next-generation trust infrastructure.

Introduction

Modern decentralized systems require governance mechanisms that engender confidence without relying on traditional centralized trust. The failures of trusted institutions in the 2008 financial crisis spurred interest in alternatives [3], and blockchain technology emerged as a **"trustless"** solution by "shifting trust from people to math" [1]. By having every network node execute the same code and verify the same ledger, blockchains enable participants to transact and coordinate without trusting a central authority [4]. However, while base-layer consensus protocols (e.g. Proof-of-Work and Proof-of-Stake) remove the need to trust single operators, higher-level governance in many systems remains off-chain or informal, often involving subjective or probabilistic trust in groups of people. For example, in Bitcoin and Ethereum, protocol upgrades are decided through rough consensus and social processes, which can lead to disputes and forks [5]. Even in blockchain communities, stakeholders must often trust that core developers or foundation leaders act in the network's best interest. This trust gap between low-level "trustless" operation and higher-level decision-making motivates the need for a **deterministic, machine-layer governance protocol** that treats governance itself as part of the formal system.

TGC v2: Machine-Layer Deterministic Trust Governance is our proposed solution to bridge this gap. "Machine-layer" indicates that all governance logic runs on the same immutable ledger and consensus layer as ordinary transactions, subject to cryptographic verification and consensus agreement by all nodes. **Deterministic trust governance** means that the rules for how decisions are made and how trust is assigned are unambiguous and encoded in advance, so that given the same initial state and inputs, all honest participants will arrive at the same governance outcomes [6] [7]. By encoding governance as a deterministic state machine, TGC v2 removes any reliance on off-chain arbitration or probabilistic outcomes. In effect, **TGC v2 replaces trusting human actors with trusting the structure and rules of the system** [1] [4]. This aligns with the ethos of "trustless trust" – achieving security via verifiable computation and distributed consensus rather than institutional guarantees [1]. All changes to the system's state (such as proposals, votes, role assignments, or trust delegations) are immutably recorded on-chain, creating an auditable trail that cannot be falsified or altered ex post [8]. Likewise, all participants can observe the system's structural state – the current set of rules, roles, and trust

relationships – in real time by inspecting the public ledger. This transparency ensures that nothing happens "behind closed doors": the entire governance process is open for scrutiny [9].

Crucially, TGC v2 introduces a formal model of **structural trust** to govern participant interactions. In contrast to reputation systems that compute a probabilistic "trust score" for participants [10] [11], TGC v2 emphasizes structural trust relationships. Participants may delegate certain powers or decision rights to others via on-chain mechanisms, and these **trust delegations** form a directed graph of accountability and authority. Because these delegations are encoded as explicit structures (e.g. who trusts whom to vote on their behalf), trust becomes an objectively defined part of the system's state, rather than a subjective or statistical estimate. The maxim **"trust is structure"** underlines that in TGC v2, **the configuration of roles, quorums, and delegations is the embodiment of trust.** For example, requiring M-of-N multi-signatures for critical actions creates a structural guarantee that no single party can act unilaterally [2]. As another example, a participant might delegate their voting power to a representative; the protocol will treat that representative's vote as binding for the delegator, which means the delegator trusts the representative by structure of the delegation graph. If that trust is abused, the delegator can revoke it – again via a structural state transition on-chain – restoring their own direct control [12]. In all cases, trust is manifested through concrete roles and rules enforced by the protocol, not by hopeful assumptions about behavior.

The remainder of this paper is structured as follows. In **Section 2 (System Model)**, we define the actors, roles, and data structures in TGC v2, along with the underlying network and security assumptions. **Section 3 (TGC v2 Protocol)** presents the governance process in detail – how proposals are created, how voting and trust delegation work, what thresholds are required for decisions, and how the system's state transitions occur. We include formal descriptions of key processes and provide tables summarizing role types and transition rules for clarity. **Section 4 (Theoretical Properties)** analyzes the protocol's properties, showing that it is deterministic and consistent, and discussing its safety (resistance to malicious actors) and liveness (ability to make progress) under certain assumptions. **Section 5 (Applications and Use Cases)** illustrates scenarios where TGC v2's approach is beneficial – from decentralized finance platforms to collaborative AI agent networks – emphasizing how immutable and transparent governance can provide real-world trust. **Section 6 (Limitations)** candidly addresses the protocol's current limitations and trade-offs, such as potential collusion issues, scalability considerations, and the need for robust participant engagement. Finally, **Section 7 (Conclusion)** summarizes our findings and suggests directions for future work in trust-centric governance. Through this comprehensive exposition, we aim to demonstrate that **TGC v2 offers a viable blueprint for governance in trustless environments by leveraging a machine-layer protocol where all participants can verify every action and trust emerges from structure rather than individual goodwill**.

System Model

Entities and Roles: The TGC v2 system involves a variety of participants, each potentially playing one or more roles in the governance process. At the most basic level, we assume a set of identifiable participants (e.g. users or nodes) each with a cryptographic identity (public/private key pair). These participants may hold governance tokens (for example, cryptocurrency stake that signifies membership or weight in voting) and may also accumulate **reputation points** as a reflection of their contributions and past behavior. In TGC v2, any token holder is a potential voter, ensuring broad inclusion akin to systems like Tezos where "anyone who holds even one token can participate in governance" [13] [14]. Participants can also act as **delegates** – a role that means other participants have entrusted them with voting power or decision authority. In a delegative (liquid democracy) model, every participant is by default their own decision-maker, but can choose a delegate to act on their behalf [15]. There is no fixed distinction between "ordinary users" and "delegates" – delegation is a dynamic relationship; a participant who

receives many delegated votes effectively becomes a representative for others, whereas a participant who delegates away their vote remains a principal who is represented by someone else. We may refer to truster (or delegator) and trustee (delegate) for a given delegation link. The delegation graph could in principle have multiple layers (A trusts B, B trusts C, etc.), but in practice TGC v2 resolves transitive delegations so that each voter's influence ultimately flows to a single active delegate or is exercised directly. Cycles in the trust graph are disallowed or automatically resolved (e.g. by preventing a direct delegation that would create a loop). All delegates must be registered participants subject to the same rules – there is no special class of "supernodes" by design, although as delegates accumulate reputation and delegated stake, they gain more influence through the structured process.

Another key role is the **validator node** (or simply validator), which is a participant responsible for running the underlying blockchain consensus and executing the governance protocol code. We assume TGC v2 is implemented on a Byzantine Fault Tolerant (BFT) distributed ledger or similar blockchain. Validators validate transactions, including governance actions, and ensure that all honest nodes maintain a consistent state. They do not have discretionary power over governance (e.g. they cannot override a passed proposal), but we include them in the model because the security of the governance mechanism inherits the security of the ledger itself. Typically, validators are selected or incentivized by a base consensus protocol (PoS staking, etc.), which is outside the scope of this paper; we assume the consensus layer provides finality and correctness for transaction ordering and execution (e.g. that >2/3 of validators are honest in a classical BFT sense, or that chain re-orgs are improbable beyond a certain depth in a probabilistic chain). Validators can also participate in governance as voters or delegates if they hold tokens or receive delegations, but they have no extra voting weight by virtue of being validators – any weight they wield comes from stake and reputation like any participant.

Reputation and Trust Scores: In addition to token holdings, TGC v2 introduces an **on-chain reputation point system** [16]. Each participant has a reputation score that quantifies their contributions, track record, and possibly community feedback within the ecosystem. Actions such as participating in governance votes, submitting useful proposals, contributing code or content, or running reliable infrastructure can earn reputation points [16]. Reputation in TGC v2 is **non-transferable and earned** – it cannot be bought or sold, only accumulated through meaningful participation. This design follows the approach of systems like Colony, where "reputation is an earned attribute reflecting the value a member brings... and is strictly non-transferable" [17]. Reputation points might also decay over time to ensure they reflect recent engagement [18], preventing someone from resting on old laurels. The purpose of reputation is to inject a notion of trustworthiness and expertise into governance: rather than a pure plutocracy where only token weight matters, TGC v2 considers reputation in determining who can do what and how much influence they wield. For example, the protocol might require a certain minimum reputation to submit a proposal or might weight votes such that both token stake and reputation contribute to voting power. An implementation strategy is a **dual-weighting** or **two-factor system**: one factor is token stake (economic weight) and the other is reputation (merit-based weight) [19]. The combination can be multiplicative or additive; one simple model is that a participant's voting power = f(stake, reputation), e.g. proportional to the product of stake amount and a function of reputation. Another model is a veto gating: proposals or votes might need a threshold in both dimensions – e.g. at least X tokens and Y reputation – to count. The exact scheme can be tuned by governance itself, but the principle is that **governance participation no longer relies solely on token weight** [16], which mitigates scenarios where a wealthy attacker with many tokens could buy control. Instead, influence must be earned by contributing to the system, aligning with the idea that active, knowledgeable members have greater say [20] [18]. The reputation system is maintained on-chain: smart contracts automatically update reputation scores based on recorded behavior, and these scores are publicly visible (with perhaps a dashboard for transparency) [16]. Additionally, TGC v2 allows **third-party protocols to query and use these reputation points** [21], meaning the reputation can become a portable credential for participants across the ecosystem (though usage outside governance is beyond our current scope).

Governance Objects: The primary objects governed in TGC v2 are **proposals**. A proposal is a structured data object (likely a transaction or smart contract call) that encodes a suggested change or action. Examples include: protocol upgrades (replacing or modifying the core code of the system), parameter changes (tuning system constants like block size limit or fee rates), treasury expenditures (allocating on-chain funds to a project), constitutional amendments (changing governance rules themselves), or any other actions the system can autonomously execute if approved. Each proposal has associated metadata: an **author** (the participant who submitted it), a **category/type**, a **timestamp or block number of submission**, and possibly a **deposit** or **bond** posted by the author (to deter spam). Proposals also have a state, such as **Draft**, **Active (Voting)**, **Passed**, **Rejected**, **Executed**, **Expired**, etc., which will be managed by the protocol state machine. The proposal lifecycle and states will be elaborated in Section 3.

We also consider **trust delegations** as first-class objects in the system state. A trust delegation can be represented as a tuple (delegator, delegatee, scope, status). The scope could be "governance voting" (default), or potentially more granular (e.g. delegate only for technical upgrade proposals but not for treasury decisions, akin to policy-specific delegation [12]). In the basic model we assume scope is broad (covering all votes) unless specified. Delegation status is active until revoked by the delegator. The set of all active delegations at a given time defines a **trust graph**, which the protocol uses to determine who effectively votes on whose behalf.

System State and Ledger: At any given time, the global state of the TGC v2 system includes: - The current **ledger** of token balances (since voting power partly derives from token stake). - The current **reputation scores** of all participants. - The set of **active trust delegations** (edges in the trust graph). - The set of **active proposals** and their states. - Any **governance parameters** (such as voting period duration, required thresholds, quorum requirements, etc. which themselves can be changed via governance).

This state is maintained on a blockchain. We assume the blockchain provides the usual properties of immutability and consistency: once a block of transactions (including governance actions) is finalized, it is append-only and cannot be altered [8]. Every participant (or at least every full node) can inspect the entire history of governance changes, ensuring **auditability**. In other words, all changes are recorded immutably and can be traced [19]. Organizational processes that in traditional governance might be opaque are here transparent and on-chain [9]. The blockchain consensus ensures that all honest nodes agree on the state (including who has what reputation, which proposals passed, etc.) and that they process state transitions deterministically in the same order [7]. Non-determinism (such as reliance on external oracles or random functions) is minimized or carefully controlled, to avoid any divergence in state.

Security Assumptions: We assume participants control their own private keys and that digital signatures are unforgeable, so actions (votes, delegations, proposals) are authentic. The underlying consensus is assumed to tolerate up to a fraction of Byzantine (malicious or faulty) validators (commonly 1/3 in BFT algorithms) without compromising liveness or correctness. We assume rational or adversarial actors may attempt to game governance – e.g. by buying tokens, creating fake identities (Sybil attacks), bribing others, or colluding. TGC v2's design includes economic and structural countermeasures to these threats. **Sybil attacks** (fake identities) are mitigated by the fact that influence is tied to costly resources (tokens that must be staked and real contributions to earn reputation) [19]. Acquiring a meaningful stake or reputation is expensive and/or time-consuming, raising the bar for Sybil attackers. **Bribery and collusion:** Because all voting is transparent, observers can detect unusual voting patterns or swings possibly indicative of bribery; however, on-chain transparency also paradoxically makes bribery easier to prove (bribers can verify how someone voted). Techniques like secret ballots or vote blinded until after an election could be considered in extensions, but in the base model votes are open. The system cannot categorically prevent collusion (no decentralized system fully can if a majority colludes), but by requiring broad quorum and multi-factor approval, TGC v2 raises the cost and coordination needed for a hostile

takeover. For instance, an attacker would need to accumulate both a large token stake and a high reputation (which requires community-recognized contributions) to have significant influence – a combination that is harder to obtain covertly. Additionally, the **immutability** of the record means any malicious decision cannot be quietly made; it will be evident to all, allowing for social or network-level countermeasures (participants could fork away if an extreme capture occurred, etc., though our aim is to minimize such scenarios through the protocol's intrinsic checks). We also assume normal network conditions where participants can communicate transactions to validators and blocks propagate within the voting periods, etc. (network delays could in theory affect timing of proposal closures, so parameters should allow comfortable margins). Finally, we assume cryptographic primitives (hashing, signatures, possible zero-knowledge proofs if used for privacy features) are secure.

In summary, the system model comprises a **decentralized ledger** with **participants** (each with tokens and reputation), a dynamic **trust graph** of delegations, and a set of **governance rules** encoded in smart contracts. This machine-mediated social contract ensures that governance follows a predictable, transparent process where every stakeholder can verify every step of decision-making. No off-chain authority is needed to enforce outcomes – if a proposal is approved by the defined rules, the blockchain will carry it out automatically or with multi-party sign-off. The next section details these rules and processes that make up the TGC v2 protocol.

TGC v2 Protocol

TGC v2's governance protocol is essentially a state machine that transitions through various phases as proposals are introduced, voted upon, and enacted. In this section, we break down the process into coherent components and provide a detailed description of the rules and algorithms at each step. We also define formulas for important calculations, such as vote tallying with trust delegations and threshold conditions for approval.

3.1 Proposal Lifecycle: A governance **proposal** undergoes a sequence of stages, enforced by the smart contract that manages governance. The typical lifecycle is: **Draft → Voting → Decision (Accepted/Rejected) → Execution (or Cancellation)**. The following describes each stage:

- **Draft/Submission:** Any participant who meets the proposal criteria can submit a new proposal. Criteria may include staking a proposal deposit (to prevent spam) and possibly having a minimum reputation score or support from a few others (to ensure seriousness). When a proposal is submitted, it is assigned a unique ID and recorded on-chain with its metadata (proposer, timestamp, content, category). At this point it can be considered in Draft status or immediately move to Voting status depending on the system's configuration. Some systems implement a **pre-voting discussion period** [22], but TGC v2 leaves discussion to off-chain forums or an on-chain discussion board module; the protocol itself is concerned with formal steps. A proposal can be canceled by the proposer before voting starts (with a penalty possibly forfeiting their deposit if they do so too late).

- **Voting Phase:** Once a proposal enters the **Voting** stage, it is open for eligible participants to cast their votes for a predefined voting period (e.g. 7 days or a certain number of blocks). During this phase, each participant's vote may be cast either **directly** or via **delegation**. If Alice has delegated her vote to Bob prior to the voting phase, then Bob's vote will automatically also count for Alice according to the delegation. Participants can typically vote **"Yes / Approve"**, **"No / Reject"**, or **"Abstain"** (neutral, counted only towards quorum but not for or against). The voting is **weighted** by each participant's voting power. Let w_i denote the voting weight of participant i. In a simple token-weighted system, w_i could be the number of tokens i has staked. In TGC v2's multi-factor

system, we define $w_i = s_i \times f(r_i)$, where s_i is the stake (token count) of i and r_i is i's reputation score, and $f(\cdot)$ is some function that converts reputation into a multiplier or additional weight. For example, $f(r)$ might be $1 + \alpha r$ for some small α, or might be piecewise (higher reputation levels unlock certain extra votes). As a concrete example for illustration, the system could assign one additional "pseudo-token" of voting weight for every 100 reputation points, up to some cap. The exact formula can be adjusted by governance itself; what matters is that w_i is computable deterministically from on-chain data at the time of voting.

Delegation in Voting: TGC v2 supports liquid democracy style delegation [15]. Formally, if participant A has delegated to B, then during voting, A does not cast a separate vote; instead, B's vote, if cast, is counted as if A also voted the same way. If B in turn delegated to C, and C votes, then A and B's weight both flow to C's vote, and so on. We define a function $D(i)$ that returns the final delegate of i: if i delegates to j, and j delegates to k, etc., eventually we reach someone who either votes directly or who has no delegate (votes directly by default). That person is $D(i)$. The weights are aggregated such that the effective weight of a voter X in the tally is $W_X = \sum_{i:D(i)=X} w_i$; i.e. the sum of weights of all participants whose delegation chain ends in X. Each such X then casts their vote (Yes/No/Abstain) and that choice is applied with weight W_X. (If a delegate does not vote at all, presumably their and their delegators' votes are simply not counted, which is effectively an abstention; rational delegates will vote on every proposal or else lose credibility.) Delegators retain the right to override: in some liquid democracy implementations, if a delegator A disagrees with their delegate B on a particular vote, A can explicitly vote on that proposal, which overrides B's decision for A's weight only. TGC v2 can allow this override: the rule is that a delegation is a default, but an explicit vote always takes precedence for that participant. Thus, delegation primarily helps those who are offline or less interested delegate to active representatives, but does not bind them absolutely if they choose to participate directly. All votes and delegations are public (unless a privacy-enhancing extension is implemented); the transparency enforces accountability – delegates know they are being entrusted and watched by their delegators, and delegators can see if their delegate voted contrary to their expectations, which might prompt changing the delegate afterward [12].

Quorum and Thresholds: TGC v2 defines for each proposal category a required **quorum** (minimum level of participation) and **approval threshold** (minimum share of votes in favor) for the proposal to be accepted. Quorum can be defined in terms of a fraction of total voting weight in the system. For instance, a system might require that at least 40% of all tokens (or weighted votes) participate in a regular proposal for it to be valid – this prevents a small cabal from passing proposals while most token holders are apathetic. The approval threshold is typically a fraction of the votes cast (or of total weight). For example, a simple majority threshold is >50% of votes in favor. More critical changes might need a supermajority, e.g. 67% or 3/4 in favor. Some systems like Tezos combine these: e.g. Tezos requires an 80% supermajority and a high quorum in certain phases [23]. In TGC v2 we generalize: each **proposal type** t has parameters quorum_t and $\mathrm{threshold}_t$. Table 1 shows an example of different categories and their requirements (note: these numbers are illustrative):

Proposal Type	Example Actions	Participation Quorum	Approval Threshold
Routine Parameter Change	Tuning gas limit or fees	20% of total weight	> 50% "Yes" (simple majority)

Proposal Type	Example Actions	Participation Quorum	Approval Threshold
Core Protocol Upgrade	Consensus or governance code change	40% of total weight	$\geq$ 2/3 “Yes” (supermajority)
Treasury Allocation	Funding a community project	15% of total weight	> 50% “Yes”
Emergency Action	Critical security patch or contract freeze	no strict quorum (time-sensitive)	$\geq$ 3/4 multi-sig approval by Emergency Council

Table 1: Example governance proposal categories and required quorums/thresholds.

These parameters ensure a balance between efficiency and safety [19]. Low-impact decisions can be made with lower turnout and simple majority, whereas high-impact or risky changes demand broad consensus. An Emergency Action is a special type allowing rapid intervention (e.g. to disable a hacked smart contract) – such actions might be entrusted to a special **Emergency Council** (a small multisig committee elected or appointed by governance) that can act quickly with a higher threshold (e.g. 3/4 of council members approving) and perhaps bypass the normal lengthy voting period. Emergency powers are limited in scope and subject to ex post review by the whole community to prevent abuse.

- **Decision & Tallying:** At the end of the voting period (say block B_{end}), the votes are tallied by the protocol. Thanks to determinism, each validator or node can independently compute the outcome by aggregating votes and delegations as described. A proposal is **Accepted** (passed) if and only if both the quorum and threshold conditions are met. For example, if it’s a Core Upgrade proposal, the protocol checks: (a) was total participating weight $\geq$ 40% of total? and (b) of that participating weight, was $\geq$ 67% in favor? If yes, the outcome is Accepted. If either condition fails (not enough participation or not enough yes-share), the outcome is **Rejected**. If Rejected, the proposal is finalized as failed, and typically cannot be re-submitted in the same form for some time (to avoid spamming failed proposals repeatedly). If Accepted, the proposal proceeds to execution.

- **Execution:** Once a proposal passes, TGC v2 leverages the blockchain’s smart contract capabilities to automatically execute the approved change whenever possible. If the proposal is a parameter change or a piece of code to be deployed, the governance contract (or associated runtime logic) will apply the changes. For instance, if the proposal was to increase the block size limit, the protocol can update that value in the chain’s parameters at a specified activation block (some systems have a delay or a separate activation vote). If the proposal is a code upgrade, in a platform like Tezos the new code is “hot-swapped” into the protocol without fork [24]. TGC v2 similarly can load and activate approved code changes. In some cases, proposals might not be fully automatable – e.g. a proposal to form a partnership or do something off-chain. Those might simply record the will of the community, and a multisig or foundation may carry it out off-chain; however, TGC v2 tries to keep governance actions on-chain to maintain determinism. Treasury spend proposals, for example, can be executed by having the on-chain treasury smart contract automatically transfer funds to the specified address if the proposal passes. This on-chain execution guarantees that outcomes of governance are enforced by code, not by trust in individuals to carry them out later [25]. In effect, the “law” (policy) is directly converted into “code” (execution). All executed changes are of course logged on-chain as transactions or state changes, so they are auditable.

- **Post-Execution and Feedback:** After execution, the proposal is marked as **Completed**. If funds were bonded by proposers or voters (as in some governance systems where voters stake tokens on outcomes), those may be returned or slashed depending on whether the proposal passed legitimately or was deemed spam/malicious. TGC v2 could implement a **slashing condition** for malicious proposals: for example, if someone submits a proposal that is clearly an attack (e.g. trying to drain treasury to themselves) and it gets overwhelmingly rejected, the proposer's deposit might be forfeited and their reputation might drop as a penalty. Conversely, successful proposers could be rewarded (Tezos rewards successful upgrade proposers with newly minted tokens [26]). Moreover, participants (and especially delegates) may see their reputation scores update based on the outcome – e.g. a delegate who consistently votes with the majority and in the best interest of the system might gain reputation, whereas one who votes for proposals that are rejected by large margins might lose reputation, reflecting the community's implicit judgment of their choices. Over time this creates a feedback loop: good governance behavior is rewarded with more trust (reputation), poor behavior erodes it. This helps align incentives: delegates want to vote responsibly to maintain their reputational clout.

3.2 Trust Delegation Rules: The trust delegation mechanism in TGC v2 deserves special attention, as it is key to the concept of structural trust. Delegation is essentially the act of trusting another participant to make decisions on one's behalf. The protocol provides a deterministic method to manage these delegations. The rules are:

- Establishing a Delegation: Participant A can delegate to participant B at any time by submitting a delegation transaction (signed by A). If A already had a delegate, this action overrides the previous one. B must not be A (no self-delegation to avoid trivial loops), and B must not be in A's downstream delegation chain (to prevent cycles). Typically, a UI or the protocol would detect a cycle attempt and reject it. The delegation can be global (covering all proposal votes) or topic-specific [12]. In TGC v2's initial design, delegations are global by default. The delegation is recorded on-chain as an active link $A \rightarrow B$. Optionally, B could be notified or might need to explicitly accept (some systems allow open delegation without consent, some require the delegate to acknowledge to prevent unwanted burdens – TGC v2 could allow either, but since delegation only gives voting power and not responsibilities, an "accept" step is not strictly necessary).

- Revoking a Delegation: A can at any time revoke their delegation, either to switch to another delegate or to revert to self-representation. This is done by a revoke transaction. Revocation is effective immediately for any future votes (if a vote is ongoing, depending on implementation, A could even override within that vote as noted). The ability to revoke at any time is crucial for accountability [12] – delegates know that if they act against A's interest, A can withdraw support. This upholds the **liquid** nature of the democracy: trust is not handed over permanently, it is continually subject to the delegator's review.

- Limits on Delegation: To prevent abuse or extreme centralization, the protocol might impose limits. For example, one participant might not be allowed to hold more than X% of the total delegated power (if they do, new delegations to them could be halted or flagged). This is not strictly necessary but could be a safeguard. Another possible rule is recursion depth limit – e.g. disallow multi-hop delegation beyond a certain depth to simplify analysis; however, since we compress transitive chains, depth doesn't matter for outcome except for potential latency of resolution.

- Interaction with Reputation: Delegation doesn't transfer reputation per se, since reputation is tied to the individual, but if a high-reputation person delegates to someone, that someone's effective influence increases (since they carry the weight of the delegator's stake and indirectly the credibility that the high-rep person trusted them – though the protocol doesn't automatically give the delegate the delegator's rep points). In some advanced designs, one could imagine trust transitivity affecting reputation – e.g. delegates who gather many delegations gain reputation as trusted community leaders. TGC v2 can incorporate a rule: **delegated trust contributes to reputation growth**. If delegate B has many incoming delegations and votes in a way deemed beneficial by those delegators (or by outcomes), B's reputation could increase faster. This creates a meritocratic cycle where good delegates rise in reputation, reinforcing structural trust in them. Conversely, if delegators revoke en masse from a delegate (perhaps due to bad actions), that could trigger a reputation decrease for the delegate, reflecting lost trust. Such dynamics must be tuned to avoid unstable oscillations, but conceptually they align with "trust is structure": the structure (who delegates to whom) directly informs the trust metrics of the system.

- Multi-Role Nodes and Delegation: In TGC's multi-layer node architecture [27], some nodes may have specialized roles (e.g. validator, council member, oracle, etc.). Delegation could potentially be role-specific. For example, one might delegate their technical governance votes to a technical committee and their economic/treasury votes to a separate financial committee. In this version (v2) we keep it simpler: any delegate effectively represents the delegator in all governance matters. Future versions or extensions could allow domain-specific trust, which is analogous to policy-specific proxies mentioned in liquid democracy literature [12].

3.3 Governance Execution and Enforcement: A cornerstone of TGC v2 is that approved proposals are executed deterministically by the machine layer. This requires an **on-chain governance smart contract** (or set of pallet modules, in a Substrate-based chain for example) that can perform privileged operations when authorized by a successful vote. Here we outline how various types of changes are enacted:

- Protocol Parameter Changes: The governance module has the authority to modify configurable parameters (stored in a central registry or state). When a parameter-change proposal passes, a transaction is issued by the governance contract to update that parameter. Since all nodes run the same governance logic, they will all update their local state accordingly. For instance, if block gas limit is a parameter and it's increased, nodes will start enforcing the new limit from the designated activation block.

- Core Code Upgrades: If the blockchain supports self-amendment (like Tezos or via built-in upgrade mechanisms), the governance proposal includes the code diff or new version. Upon approval, either the network automatically switches to the new code at a chosen block, or a multi-signature group (like a hard fork coordinator committee) is triggered to deploy the new code. However, the latter introduces some off-chain trust; ideally the chain itself can load new logic. In a platform like Substrate, governance could schedule a runtime upgrade by storing the new Wasm blob on-chain and instructing nodes to replace the runtime (this is exactly how Polkadot/Kusama governance works, fully on-chain). TGC v2 would leverage similar capabilities – making it a truly **self-governing chain**. All participants, including validators, then follow the new rules after upgrade, with no manual intervention needed, fulfilling the vision of automatic rule enforcement.

- Treasury/Financial Actions: A treasury is often a smart contract holding funds. TGC v2's treasury proposals specify a recipient address and amount (and reason metadata). If passed, the governance contract executes a token transfer from the treasury to the recipient. This requires that the governance contract (or a designated multi-sig of council) has spending rights. We prefer

to avoid any manual step, so likely the governance logic itself is coded to have conditional permission to move funds when proposals pass.

- Elections of Roles: The governance process can also be used to elect people to certain roles (e.g. an oversight council or emergency committee). An election could be structured as a special proposal listing candidates, or a separate voting mechanism altogether (like Quadratic voting or score voting). For deterministic execution, once the voting period for an election ends, the contract determines the winners (e.g. top N vote-getters) and updates the role assignments in state. Those assignments (like "Alice is now on Security Council") become part of the system state that others can inspect. Role assignments might come with specific permissions (for instance, members of an emergency council might be able to trigger an emergency pause). The history of these elections is recorded, and terms could be finite (requiring periodic re-election). All this is managed by on-chain logic, eliminating back-room appointments.

- Multi-Signature Enforcement: There are certain actions where instead of one proposal directly doing a thing, multiple approvals are needed in sequence. For example, if a drastic measure is to be taken, the design might require two successive votes (like Tezos' multi-phase voting [28]). TGC v2 supports multi-step processes by encoding stateful proposals: a proposal can trigger a follow-up proposal automatically. For instance, an **Amendment** may require an **Exploration Vote** then a **Confirmation Vote**; the system will move it through those states and only execute if it passes both. This ensures structural rigor for big changes. Another scenario is requiring both the general token holders and a specialized committee to approve something (bicameral governance). That could be modeled as two proposals in parallel or a requirement that two distinct groups reach agreement. The protocol can be configured to check multiple tallies.

3.4 Formalization of Voting Outcome: We present a formal condition for proposal acceptance in TGC v2. Consider a proposal P of type t. Let V_P^+ be the set of participants (or effectively delegates) who voted "Yes" on P, and V_P^- those who voted "No". (Abstentions are excluded from both sets but count towards quorum.) Define:

- Total weight participating $W_{\text{part}} = \sum_{i \in V_P^+ \cup V_P^- \cup V_P^0} w_i$, where V_P^0 is abstainers (if abstain counts as participating) or we include abstainers in W_{part} if counting toward quorum.
- Yes weight $W_{yes} = \sum_{i \in V_P^+} w_i$.
- No weight $W_{no} = \sum_{i \in V_P^-} w_i$.

Also let W_{total} be the total possible voting weight in the system (e.g. sum of all staked tokens, possibly plus any rep weight if applicable).

The **quorum condition** is $W_{\text{part}} \geq \text{quorum}_t \times W_{\text{total}}$. For example, if quorum_t = 0.20 (20%), then at least 20% of all weight must vote (including abstain if counted).

The **approval condition** is $\frac{W_{yes}}{W_{yes}+W_{no}} \geq \text{threshold}_t$. If abstains are counted in denominator or not depends on design; typically abstain is neutral and not counted in yes/no percentage (only in quorum). We assume threshold is on yes vs no only.

A proposal P is **Accepted** iff both conditions hold; otherwise **Rejected**.

This simple formula is evaluated by the contract at tally finalization. Determinism is guaranteed because w_i and all votes are known values on-chain at that time, so every node will compute the same result.

3.5 Example Scenario: To illustrate the protocol mechanics end-to-end, consider a scenario within a decentralized organization using TGC v2:

- A developer member, Alice (who has moderate reputation and some tokens), submits a proposal to upgrade the network's smart contract runtime to add new features. The proposal is categorized as a Core Protocol Upgrade, which per Table 1 requires a 2/3 supermajority and 40% quorum. Alice stakes 100 tokens as a deposit for the proposal. The proposal is posted on-chain at block 1000.

- Bob, a well-known community expert with high reputation (and a token holder), is a delegate for a few others. Charlie and Diana have delegated their votes to Bob because they trust his technical judgment.

- The voting period runs for two weeks. All token holders (or their delegates) can vote. Bob reviews Alice's code and likes it; he votes "Yes". Because Charlie and Diana delegated to Bob, Bob's vote automatically counts their weights as well. Bob himself has 500 tokens and high rep, Charlie has 300 tokens, Diana 200, so assume Bob's effective voting weight W_Bob = (500 + 300 + 200) * f(rep) – if Bob has some rep multiplier, maybe it's effectively ~1100 tokens weight combined.

- Erin, another participant, does not delegate and votes "No" with her 400 tokens (she's skeptical of the upgrade). A number of other participants vote; by the end, let's say 10000 tokens worth of weight participated out of 20000 total (50% quorum achieved, above 40% required). Of those, 7200 worth voted Yes, 2800 No. That's 72% Yes which exceeds the 67% threshold. Thus the proposal meets both quorum and supermajority criteria.

- The proposal is marked **Accepted** on-chain. After a short delay for finality, the new code contained in the proposal is automatically deployed. At block 1100, all nodes switch to the upgraded runtime as dictated by the proposal. The chain now runs the new version.

- Alice's deposit is returned (perhaps even with a reward), and Bob, having been on the winning side, gains a small reputation boost for supporting a successful change. Erin's vote was in minority; no penalty, but perhaps no rep gain either. Charlie and Diana, who delegated, see that their delegate Bob's decision aligned with the majority – their trust in him is reinforced. Had the vote gone differently (if Bob voted against their interest), they always have the option to revoke delegation for future.

- The entire sequence – proposal content, votes cast (who voted yes/no and their weights), and the final execution – is recorded on-chain. Any observer or auditor can later verify that the rules were correctly followed: the voting tallies met the required threshold and the code that was activated matches the proposal. The transparency builds confidence that the upgrade was not malicious or sneaked in; it was the result of a clear structure of trust and voting.

This scenario highlights how **deterministic, structural trust governance** operates in practice: no single individual could trigger the change (Alice alone couldn't, Bob alone couldn't). It required a structured collective action recorded immutably. Everyone's influence was according to predefined rules (token and rep weight, plus any delegations). The outcome was automatically enforced, removing any ambiguity about whether the community's decision would be honored. Participants like Charlie and Diana did not have to personally analyze the code; they relied on the structure (delegation to Bob, who is accountable to them) rather than blind trust – they could withdraw their trust if needed.

3.6 Additional Features: TGC v2 can incorporate further processes to enhance robustness:

- Periodic Governance Cycles: The system might operate in fixed cycles (e.g. monthly governance cycles) where proposals are collected and voted in batches. This can give a regular cadence and avoid continuous voting fatigue. During each cycle, multiple proposals can be voted on, possibly with a limit on how many can pass to prevent overwhelming changes at once.

- Community Initiatives and Referenda: In addition to protocol changes, TGC v2 can handle community referenda – essentially sentiment surveys or signaling votes that don't automatically execute changes but record an official stance. The process is similar (voting, threshold) but outcome is not an execution, just a resolution (e.g. "the community endorses principle X"). This can be useful for decisions that involve off-chain follow-up.

- Dispute Resolution: In case of contentious decisions, TGC v2 might include a mechanism to delay execution if a certain minority strongly objects. For example, if >25% vote No, maybe trigger a second discussion period before finalizing (to try reaching more consensus). This is a design choice – some systems have multiple rounds to resolve disputes [28].

- Upgrade of Governance Itself: Since this is a v2 protocol, presumably evolving from a v1, we note that the protocol allows **meta-governance** – changes to the governance rules via the governance process. For instance, the community could decide to adjust the quorum percentages, or incorporate quadratic voting, etc., by proposing amendments. This reflexivity means the governance system can adapt over time, ideally without hard forking. TGC v2's design tries to bootstrap with sensible defaults (based on lessons from other DAO governance models) but not to ossify them.

Having described the mechanics, we now turn to analyzing why this protocol has strong trust properties and what guarantees we can make about its behavior.

Theoretical Properties

In this section, we discuss the theoretical properties of the TGC v2 protocol, focusing on how it ensures determinism, security, transparency, and other desirable attributes. We ground the discussion in established principles from distributed systems and game theory as appropriate, citing proofs or precedents where possible.

4.1 Determinism and Consistency: TGC v2 is fundamentally a deterministic state machine. This means that the sequence of state transitions (proposal submissions, voting, tally, etc.) is fully predetermined by the rules given the inputs. All nodes running the protocol will compute the same result for each proposal's outcome, as long as they start from the same prior state and see the same votes. This property is critical for a trustless consensus environment [6] [7] – it eliminates ambiguity and the possibility of inconsistent governance outcomes on different nodes. In practice, determinism is ensured by avoiding any dependence on off-chain data or random choices in the governance logic. For example, if multiple proposals conflict (say two proposals both try to change the same parameter differently), the protocol should deterministically handle it (e.g. by priority rules or only one can execute, others auto-fail). By predefining such conflict resolution, we avoid nondeterministic forks in governance state.

Every governance action is processed in a transaction that is totally ordered via the base consensus, ensuring a single canonical history. If two participants submit proposals at the same time that exceed

some limit, one will be included first by the ledger order and the other might be queued or rejected per rules – but all nodes agree on which happened first. Thus, **consistency** is maintained: at any block height, the set of "passed proposals" is the same for everyone, and thus the set of active rules is the same. This is analogous to state machine replication: governance logic is part of the replicated state.

4.2 Immutability and Auditability: By building on a blockchain, TGC v2 inherits immutability – once a vote or delegation is recorded in a finalized block, it cannot be altered or removed without violating the chain's integrity. This means the governance history is **tamper-proof** [8]. Immutability is crucial for audit and trust: participants can trust that the record of decisions is correct and complete (absent a 51% attack on the chain, which we assume away in normal operation). **Auditability** implies that any third party can come later and verify that a certain proposal passed with X votes, that a certain delegate had Y weight, etc., by replaying the transactions. This verifiability fosters confidence in the system's legitimacy [4]. In traditional organizations, auditing governance (like verifying a vote count in a shareholder meeting) requires trusting the scrutineers or complicated paper trails; here, it's trivial for software to recount and confirm any vote. Additionally, because all structural states (like who delegated to whom, and current parameter values) are on-chain, participants can audit the current power distribution – for example, one can compute which delegates hold the most aggregate weight, or check that no role appointments violated the set process.

4.3 Transparency: Transparency in TGC v2 is maximal: all participants can observe all structural states. The current delegate graph, reputation scores, vote counts on active proposals, and results of past proposals are all openly accessible. As the Algorand Foundation notes, on-chain governance processes are "transparent and open for anyone to scrutinize," unlike closed-door governance [9]. This transparency has two effects: (1) **Accountability:** Delegates and decision-makers know that their actions are public, which discourages rash or self-serving decisions because they can be held accountable by the community (delegators can withdraw support, for instance). (2) **Trust through visibility:** Stakeholders do not need to blindly trust that the rules are being followed – they can see it. The old adage "don't trust, verify" is realized; e.g., if a proposal passes, anyone can verify the threshold was indeed met according to the known rules. The state of trust relationships is also visible, which is a double-edged sword (it enables analysis of the power structure, but also, as mentioned, could enable targeted bribery or pressure on major delegates; this is addressed under limitations). Nonetheless, the guiding philosophy is that sunlight is the best disinfectant: problems or concentrations of power can be identified and addressed by the community precisely because the structure is transparent, whereas hidden influence would be more pernicious.

4.4 Security and Resistance to Takeover: We analyze the protocol's ability to resist malicious governance takeovers. In a purely token-weighted system, an attacker who acquires >50% of tokens (by purchase or theft) could push through arbitrary proposals, potentially damaging the system (a classic 51% attack on governance). TGC v2's multi-factor approach raises the bar significantly. With **dual trust factors** (stake and reputation) [29], an attacker would need to not only buy tokens but also build a credible reputation to have full influence. Reputation, being non-transferable and earned over time, is harder to amass quickly or by spending money [17] [18]. This provides a form of **Sybil-resistance and plutocracy-resistance**: you can't instantly gain high voting power without community trust. Even if an attacker got many tokens, they might be limited by low reputation from making major changes (for example, the system could say that critical proposals require the proposer to have, say, top 10% reputation or something – an attacker wouldn't meet that). Moreover, delegations tend to concentrate on trustworthy figures; a new unknown entity wouldn't immediately get others delegating to them.

Structurally, requiring **quorum** protects against sneaky attacks when many are inactive. For instance, if only 10% of voters are paying attention, a malicious actor with 9% support could pass something in a naive system; but if quorum is 20%, the proposal would fail due to low turnout, forcing the attacker to

involve more of the community (which likely draws scrutiny). The requirement of **supermajorities** for critical decisions (like 2/3 or higher) means that even if an attacker obtains a simple majority, they still cannot ram through a protocol overhaul – they'd need an overwhelming level of control, which is correspondingly harder to achieve without detection.

TGC v2 also benefits from distribution of control via **structural multi-party mechanisms**. For example, having an **Emergency Council** for fast moves with limited scope ensures that not even a majority of token holders can, say, pause the chain on a whim – it's gated by a small expert group's approval, but that group is itself elected and can be removed by the larger governance if it misbehaves. Conversely, the council alone cannot pass non-emergency changes – that needs the broader vote. This segregation of duties (like separation of powers) increases security by not putting all eggs in one basket. It becomes somewhat like a bicameral legislature: two different bodies (general token holder vote and special council) both have to agree in their respective domains.

One can attempt to quantify security in terms of the fraction of colluding participants needed to compromise the system. In TGC v2, suppose an attacker tries to pass a malicious core upgrade to seize funds. They would need to satisfy the threshold in that category – e.g. 67% of weighted votes. That means controlling >67% of the effective voting weight. If reputation weight is significant, controlling that much weight might entail controlling >67% of tokens and having comparable reputation. If reputation is roughly correlated with sincere participation, an attacker controlling that much rep would have to have deeply infiltrated the community or been a long-time contributor (at which point they're not really an outside attacker anymore, or if they turned malicious it would be an extraordinary betrayal that likely would have other consequences). Thus, the system is secure under the assumption that less than a supermajority of the community can be corrupt or coerced simultaneously – a standard assumption for distributed consensus and governance (often one assumes >1/3 honest for blocking power, etc.). We can state: **Safety**: No invalid or adversarial proposal will be adopted as long as the set of rational, honest participants controlling more than (1 – threshold) fraction of voting weight coordinate to vote against it. For threshold 2/3, this means if at least 34% by weight are strongly against a malicious change, they can ensure it doesn't happen. This parallels Byzantine fault tolerance logic (where 1/3 can block if 2/3 needed to confirm).

Additionally, because **all nodes verify transactions** and the protocol's execution, any attempt to execute something that wasn't approved (e.g. a rogue validator tries to trigger an execution without proper vote) would be rejected by other nodes as an invalid state transition. So the system is **byzantine resilient at the protocol level** – it does not rely on a single server or admin to enforce governance; the network collectively enforces it, and any deviation is a consensus fault.

4.5 "Trust is Structure" Formalized: The slogan "trust is structure" can be given a formal interpretation: In TGC v2, any reliance on a participant's actions is always mediated by a structural constraint or relation that is verifiable by others. For example, if I (as a user) "trust" a delegate to vote wisely on my behalf, I am not simply hoping they do – I have structurally bound my vote to theirs, and I (and everyone) can verify how they vote. This is an enforceable form of trust – if they fail me, I can structurally withdraw trust. Another example: multi-signature requirement for treasury spending means the community doesn't have to trust a single treasurer; the structure enforces that at least M of N specified people (perhaps each elected from different stakeholder groups) sign off [2]. Thus, the system transforms what would be a matter of personal trust ("I trust the treasurer won't steal funds") into a matter of structural constraint ("funds cannot be moved unless 4 of 5 independent keys approve, which makes theft extremely unlikely without broad collusion"). Trust is thereby factored into the architecture. A theoretical property here is **no single point of failure in governance**: every decision of consequence requires either consensus of a majority or a predetermined committee, preventing unilateral control. In graph terms, the governance power is decentralized – ideally, well-distributed among many participants and not highly centralized. We

can measure the centralization of power via something like the Herfindahl index of delegate vote shares or the Gini coefficient. While these can fluctuate, the design's inclusion of reputation and ease of delegation aims to encourage a flatter power distribution (people will tend to delegate to those with demonstrated expertise, but if one person accumulates too much, others might become cautious or alternative delegates rise, etc.).

4.6 Liveness and Adaptability: Beyond safety, we consider **liveness** – the ability of the system to continue making decisions and evolving. Because TGC v2 is deterministic and has fixed voting periods, one might wonder, could governance grind to a halt if there's too much disagreement or not enough participation? The quorum mechanism does mean a proposal fails if not enough show up. However, over time the community can adjust quorums if they prove too high to meet regularly. The system can include an adaptable quorum rule (some systems like Polkadot have adaptive quorum biasing: lower thresholds if turnout is high, etc., but that's complex). For simplicity, assume quorums are tuned and perhaps gradually lowered if consistently not met (via an amendment). As long as some threshold of honest or interested participants are engaged, proposals that have broad support should pass. If the community is utterly apathetic (e.g. <20% ever participate), then either nothing major changes (status quo holds, which might be fine until interest picks up), or changes are made to re-engage (like incentive tweaks).

TGC v2 ensures at least that anyone can initiate a proposal (given minimal criteria) – so it's not bottlenecked by a central committee deciding the agenda. This openness fosters liveness: if an upgrade is needed, any developer can propose it. There is no gatekeeper to stymie needed evolution (unlike off-chain governance where a core dev team could ignore community wishes). The only case where liveness fails is if the required thresholds are unreachable because of fragmentation or voter apathy – a known challenge in governance. Mitigations include: incentives for voting (perhaps token rewards for voters or slashing for non-participation in extreme cases), delegation to concentrate votes (so even if many don't care, they delegate to those who do, thus effectively raising participation). The liquid delegation system is itself a liveness booster: by allowing dynamic representation, it "ensures decisions are made by people with the most stake or expertise" even if others step back [30]. As one source notes, "liquid democracy provides flexibility – a voter may delegate their vote when busy but participate directly when able" [31], thus the system can always adapt to the level of engagement individuals can afford.

Adaptability also refers to the ease of changing the rules. TGC v2 being self-governed means it can adapt its own constitution in response to unforeseen challenges (with sufficient consensus). This meta-adaptability is a hallmark of upgradable chains like Tezos (often touted as self-amending [32]). It future-proofs the system to an extent: rather than being locked into the initial design (which might have flaws), the community can iteratively improve it.

4.7 Incentive Alignment: While not a formal consensus property, it's worth noting game-theoretic considerations. TGC v2 tries to align incentives such that honest participation is rewarded and malicious behavior is penalized. Through reputation gains for positive contributions and potential losses for negative, participants have a rational incentive to build trust and not betray it. Delegates who accumulate a lot of voting power essentially hold a position akin to elected officials; they have reputational capital at stake. If they violate their constituents' trust (say a delegate flips to support a harmful proposal for a bribe), they risk immediate loss of delegation (which can be seen on-chain the moment people revoke) and long-term loss of reputation, reducing their future influence. Meanwhile, an honest delegate who consistently acts in good faith may attract more delegations and perhaps direct token rewards (some systems reward delegates with a portion of inflation or so for their service). The net effect is to encourage a stable equilibrium where a diverse set of competent delegates manages routine decisions, and broad community input kicks in for major decisions – roughly analogous to how representative democracies with recall mechanisms can be stable.

One could formalize aspects of this as a repeated game: participants care about their reputation which affects future utility (influence, rewards), so even self-interested ones will behave well to maximize long-term gain, as long as the system reliably measures and applies reputation changes. This is similar to how blockchain miners/validators are incentivized by future block rewards to not misbehave in the present; here the reward is continued trust.

4.8 Comparisons and Formal Guarantees: For additional context, we compare TGC v2's governance determinism to the traditional approach. In off-chain governance, outcomes can be ambiguous (there might be multiple competing "official" results if factions disagree, leading to splits). By contrast, TGC v2 yields a single source-of-truth for governance decisions [33]. This single truth is what allows the system to be called a **"confidence machine"**: users can have high confidence in the system's rules being applied as written, rather than trusting persons [34] [4]. The formal guarantee akin to a mathematical theorem we can state is:

- Theorem (Deterministic Finality of Governance): Assuming the underlying blockchain achieves consensus finality on blocks, any governance decision (proposal pass/fail) determined by TGC v2's rules is final and agreed upon by all honest nodes with the same certainty as a transaction finality. In other words, once a proposal is marked **Passed** and executed on-chain, it is irrevocable and not subject to later dispute; conversely, if it failed, no alternate interpretation can claim it passed. This is a deterministic finality, stronger than the probabilistic finality of some PoW blockchains. It moves governance into the same trust domain as transaction execution.

- Corollary: The state of the system (including trust relationships and roles) is always consistent and globally observable. There is no eventual consistency issue; the state is as up-to-date as the latest block. If two users query who Alice's delegate is, or what the current protocol version is, they will get the same answer (assuming they are on the same canonical chain). This is trivial, but worth noting in contrast to systems where state might be subjective or require querying some off-chain registry.

In summary, the theoretical underpinnings of TGC v2 show that by encoding governance in the blockchain, we achieve **determinism, immutability, and transparency**, and by incorporating a sophisticated trust structure (delegations + reputation + multi-sig checks), we enhance **security and trustworthiness** of the governance outcomes. The next section will illustrate how these properties enable practical applications that would be difficult to manage under less structured or less deterministic governance systems.

Applications and Use Cases

The TGC v2 protocol's features make it suitable for a wide range of applications where trust, transparency, and decentralized control are paramount. Here, we outline several use cases, demonstrating how TGC v2 can be deployed to improve governance and coordination in different domains.

5.1 Decentralized Autonomous Organizations (DAOs): Perhaps the most direct application is in **DAO governance**. DAOs are member-managed organizations (often for managing a shared treasury or protocol) that operate via smart contracts. Many DAOs today use basic token voting for decisions, sometimes off-chain votes recorded via snapshots and then manually executed. By adopting TGC v2, a DAO can achieve fully on-chain governance where every proposal is transparently voted on and automatically executed, eliminating the need for multisig signers to carry out the community's decisions. For instance, consider a DeFi protocol DAO governing parameters like interest rates or adding new

features. With TGC v2, the DAO could allow any token holder with a certain reputation (perhaps earned by providing liquidity or contributing code) to propose changes. Those proposals are voted on by the community, with perhaps large token holders delegating to domain experts for technical proposals. The results are enacted on-chain, e.g. changing an interest rate in the lending smart contract, all in one seamless process. This greatly reduces governance friction and increases stakeholder confidence – the rules of the protocol are truly controlled by its users in a verifiable way. Furthermore, because TGC v2 supports reputation and dual-weight voting [16] [20], it addresses common DAO governance concerns: for example, it helps mitigate plutocracy (where a whale with lots of tokens could otherwise dominate votes) by giving weight to community contributors. A concrete example: say a developer with minimal tokens but who built most of the project has high reputation – their vote can carry significant weight, ensuring decisions aren't solely driven by passive capital. The **transparency** of TGC v2 also helps DAOs comply with any emerging regulatory expectations or internal policies: every treasury movement is auditable, every decision rationale is recorded, which is key for things like insurance DAOs, or charity DAOs where accountability is important.

5.2 Self-Governing Blockchain Networks: Blockchain platforms themselves require governance (often referred to as on-chain governance). TGC v2 can serve as the governance layer for blockchains (much like Polkadot's OpenGov or Tezos' governance). The use case is the network that can upgrade and configure itself without hard forks. By implementing TGC v2 as the native governance module of a blockchain, the chain's users (typically token stakers) can vote on core protocol upgrades, parameter tweaks, and economic adjustments in a way that is **fork-free and deterministic**. For example, a proof-of-stake blockchain might use TGC v2 to manage its staking parameters (inflation rate, validator count), where an upgrade proposal is posted by any core developer or token holder with sufficient backing, then all validators and holders vote. The deterministic governance ensures that if it passes, the new runtime code is automatically adopted at a certain epoch – a process similar to Tezos which "allows the system to coordinate the selection of new updates through popular voting and integrate them automatically" [32] [14]. This provides the network with adaptability (it can evolve quickly with community consent) and avoids contentious splits (since there's a formal mechanism to resolve differences). In practice, networks like **TrueGuardChain (TGC)** itself – presumably the blockchain behind the True Guard Coin project – could implement TGC v2 to govern not only the on-chain modules but also to orchestrate its multi-layer validator network and AI integration parameters. Because all structural states are visible, participants can monitor the health of the governance (like how decentralized is the vote turnout, etc.) and make adjustments to improve it.

5.3 Collaborative AI Governance: The "AI era" motivation behind TGC hints at use in governing **AI systems or agent networks** [35]. As AI agents become more autonomous and interact in networks, there is a question: how do we trust AI decisions and ensure alignment with human values? TGC v2 could be leveraged to create a **deterministic governance layer for AI collectives**. For example, imagine a network of AI-driven or AI-assisted services (like decentralized AI cloud or an AI-managed investment fund). We could instantiate a governance DAO comprising both humans and AI agents as participants. TGC v2's trust structure allows human stakeholders to retain ultimate control by requiring that any critical action an AI proposes (say reallocating a large portion of funds) goes through a governance proposal where humans (or a human-AI council) vote. Every change the AI system wants to implement is logged and needs approval, making the AI's "thinking" transparent and verifiable. Additionally, AI agents themselves could be given a sort of reputation score in the governance system (based on performance or reliability) and more trusted agents could be delegated certain routine decisions. Because TGC v2 is machine-readable and enforceable, one can encode constraints for AI: e.g. an AI model can only be deployed if X% of trusted auditors (could include other AIs or humans) vote that it passed safety checks. This is essentially applying structural trust to AI governance: not trusting an AI output blindly, but wrapping it in a protocol that ensures multiple eyes and consensus on it. Another angle: consider federated learning or a network of AI model trainers – they could use TGC v2 to vote on model updates or parameter tuning in a decentralized

learning process, ensuring no single data center's model update is accepted unless a certain quality threshold consensus is reached by the others, etc. This could improve trust in the model because it's like a "constitutional" process for the AI's evolution.

5.4 Supply Chain and Consortium Governance: In multi-stakeholder supply chains or consortia, no single entity is fully trusted to make decisions. A blockchain is often proposed to improve transparency for supply chain data, but governance of the chain or network itself remains an issue. TGC v2 could be used to allow all stakeholders (manufacturers, suppliers, distributors, regulators, even consumers represented indirectly) to participate in governance of a supply chain network. For example, consider a consortium that oversees an **organic food certification ledger**. They can use TGC v2 such that rule changes (e.g. what qualifies as organic, who is allowed to issue certificates) are decided by voting among accredited members. If the consortium includes various companies, each might have votes weighted by some stake or reputation (maybe based on their history of compliance). All changes to the network's rules or membership could be handled transparently on-chain rather than closed board meetings. This ensures buy-in and trust: because all participants see that decisions are made fairly and no single member can covertly change a rule. The immutability is appealing to regulators (who could audit that no tampering with records or rules occurred). Similarly, any **joint venture or alliance** (like cross-border trade networks, energy grids, etc.) where autonomy and trust are needed can benefit. Essentially, TGC v2 provides a digital constitution for consortia – the rules of collaboration and decision-making encoded so that no single member's betrayal can upend the system without others noticing and being able to respond.

5.5 Public Sector and E-Governance: In the public sphere, one can envision local governments or organizations using a blockchain governance system for certain decisions to increase transparency and citizen trust. For example, a city could use a TGC v2-like process for participatory budgeting: citizens (or their delegates, perhaps elected councilors) vote on budget proposals, with every vote recorded publicly, ensuring transparency in how funds are allocated. Reputation could be given to citizens who regularly participate in community projects, to weight their say more (this is a bit controversial in one-person-one-vote contexts, but for participatory processes it might highlight those with expertise). Another public use: **University governance** (where faculty, students, admin form a senate). A permissioned version of TGC v2 could let the university's stakeholders vote on certain policies, with weights and delegations (e.g. departments delegate to reps). The immutable record would reduce disputes ("the vote was rigged!") because it's auditable by all. We can even imagine political **liquid democracy** experiments – some communities have considered letting citizens delegate their votes on issues to representatives dynamically. TGC v2 is essentially a robust infrastructure for liquid democracy [15], with the benefit that it's secure and transparent. A city or online community could pilot this for non-binding resolutions or even binding ones if laws permit. The "trust is structure" principle might be very attractive to populations who currently distrust institutions – knowing that no backroom deals can change outcomes could improve confidence in democratic processes.

5.6 Web3 Platforms and Identity Systems: Web3 often involves decentralized identity and reputation (e.g. DID standards, trust networks for verifying credentials). TGC v2 could integrate with an identity system where identity providers and verifiers themselves are governed by such a protocol. For example, the **Trust over IP Stack** proposes governance frameworks at each layer [36] [37]. TGC v2 could implement the machine layer governance for a DID network: stakeholders (maybe issuers, holders, verifiers) vote on schema changes, acceptance of new members (like adding a new root CA), etc. Because identity trust is sensitive, a deterministic governance ensures no single company can arbitrarily change how identities are handled. It formalizes trust decisions – e.g. cross-certifying a new identity provider might require a supermajority of existing trusted providers, recorded immutably (similar to how web-of-trust requires multiple endorsements, but here it's governed by policy on-chain). This could prevent incidents like rogue certificate authorities in WebPKI, by requiring structural consensus for trust.

5.7 Open Source Project Management: Open source software projects (especially large ones like Linux, or decentralized protocol projects) sometimes struggle with governance (who decides on new features or directions?). Typically a benevolent dictator or a core committee does it, which is a soft trust model. If an open source project wanted to decentralize that, they could use TGC v2: contributors earn reputation through code contributions, users hold tokens (maybe non-financial tokens allocated to those who use/need the project), and proposals for major changes are voted on. This might be ideal for blockchain-based open source where tokens exist anyway. But even without tokens, one could adapt the weighting to purely rep-based (like one contributor-one vote weighted by contributions). The main gain is transparency and preventing forks through disagreement by having a clear formal process.

In all these applications, the core values TGC v2 emphasizes are key selling points: **every change is recorded immutably**, **everyone can observe the state and progress of governance**, and **trust arises from the enforced structure** rather than any individual's promise. This makes systems built on TGC v2 attractive wherever mistrust or lack of accountability have traditionally been pain points.

We can cite a relevant example: consider the **Compound finance protocol** (a DeFi lending platform) which has a governance token COMP. They have a governance module (Governor Bravo from OpenZeppelin) which allows token-weighted voting and on-chain execution of parameter changes. Compound's success shows that even billions of dollars can be managed by on-chain votes – however, early on it faced issues like a proposal passing with low turnout. TGC v2's enhancements (reputation, quorum, delegation) would mitigate such issues, leading to more robust outcomes. Another example, **Gitcoin DAO** moved to a delegated model with a Steward Council to improve decision quality [38] ; that's akin to what TGC v2 does systematically (mixing direct and representative democracy). The use cases therefore span decentralized finance, decentralized governance of blockchain networks, and even beyond crypto into any multi-stakeholder decision system that can benefit from cryptographic guarantees.

Limitations

While TGC v2 provides a comprehensive framework for deterministic trust governance, it is not a panacea. We must acknowledge several limitations and challenges, both inherent to the approach and those arising in practical deployment:

6.1 Complexity and Usability: The **sophisticated ruleset** of TGC v2 – involving stake, reputation, delegation graphs, multiple proposal types with different thresholds – introduces complexity that can be daunting to users. In contrast to a simple majority vote, participants need to understand concepts like how reputation impacts their vote, how to delegate and when to override a delegate, what the various quorum requirements are, etc. This learning curve could discourage participation or lead to mistakes. For example, a user might delegate and forget about governance, not realizing their delegate changed stance. Or voters might not fully grasp the consequence of abstaining versus not voting. Additionally, the user interfaces must convey this information clearly (e.g., showing the user "You have 100 tokens and 50 rep, so your direct vote weight is X. You delegated to Alice, who currently has Y total weight including yours."). Without excellent UI/UX, the theoretical transparency doesn't automatically translate to intuitive transparency for a non-expert user. This is a known issue in DAO governance – participation is often low because many find the process too confusing or time-consuming. TGC v2 potentially exacerbates that by adding more moving parts (though features like delegation are meant to alleviate involvement for those who opt out). A possible mitigation is progressive disclosure: not all users need to think about all rules; casual users might just pick a delegate and trust them, power users engage in proposals. Still, complexity remains a limitation.

6.2 Participation and Voter Apathy: A governance system is only as good as the engagement it receives. **Quorum requirements** can protect against low-turnout decisions, but they can also lead to **decision deadlock** if apathy is widespread (no proposals meet quorum). Particularly in blockchain contexts, many token holders are passive or purely speculative and do not vote. If reputation incentives or rewards are not strong enough to motivate participation, the system might struggle to reach quorums. Delegation helps by concentrating votes in active hands, but it relies on users actually delegating. Some token holders might not even bother to delegate (or even know they should). As a limitation, if only a small fraction of users effectively control governance (even if by delegation), one might question how truly decentralized or representative it is in practice. For instance, in many DAOs, <10% of tokens vote regularly. TGC v2's design tries to counteract that by giving more weight to engaged users (via rep) and punishing non-participation indirectly (rep decay), but it cannot force people to care. There is also a risk that **excessive governance demands** (too many proposals, too frequent voting) lead to fatigue. If the protocol is successful and widely used, dozens of proposals might be up for vote at any time. Participants could be overwhelmed, again leading to reliance on delegates. That in itself is fine, but it means effectively a smaller group of delegates have a lot of power – which while earned, could concentrate decision-making (it starts looking more like a traditional elected council). In worst case, if participants become too disengaged, the system could be captured by a motivated minority who always turn out to vote while others don't (though reputation being needed too somewhat raises that bar).

6.3 Collusion, Bribery and Coercion: TGC v2 is transparent, which is great for accountability, but it also means votes are **public**. This enables **vote buying or coercion**: an attacker can promise to pay tokens to certain voters/delegates if they vote Yes on proposal P, and can verify from the record if they did. This is a classic problem in open voting systems. In traditional elections, secret ballots are used to mitigate vote buying – in on-chain governance, implementing secrecy is possible (with cryptographic schemes), but TGC v2 as described does not include secret voting. Therefore, **bribery markets** or "governance mining" could arise, where people essentially sell their votes or delegation rights. Delegates, especially, could be lobbied or even internally collude with each other to form an oligarchy. For example, imagine five top delegates each controlling 10% of vote weight. They could agree to all support each other's proposals and form a bloc that basically decides everything (this is somewhat akin to a political party or cartel). The reputation system might catch if they are acting against the wider interest (maybe others would downvote them or revoke delegation), but it might also be that they justify their moves convincingly and maintain rep while still collectively dominating. This is a limitation because **structural trust does not eliminate the possibility of human arrangements outside the structure** – it only ensures that actions go through the structure. People can still coordinate off-chain in malicious ways. There's also the scenario of **coercion**: a powerful entity (government or hacker) could threaten delegates or large voters to vote a certain way, and they can prove compliance via the public vote record. This is again the flip side of transparency.

6.4 Majority Tyranny and Minority Protection: The protocol, like most democratic designs, ultimately trusts majority (or supermajority) decisions. There is a risk of **tyranny of the majority** where the structure could be used to systematically disadvantage a minority. For example, consider a blockchain where a majority of token holders from one region could pass a proposal that censors or penalizes another region's users. The minority might have no recourse if outvoted (except leaving the system). Traditional systems often have constitutions or rights that even a majority cannot override easily. TGC v2 currently has no explicit notion of fundamental rights or safeguards beyond the thresholds and perhaps an emergency council. One could encode special rules (like "a proposal that removes the voting rights of an entire group is not allowed"), but generally any rule is changeable by some supermajority. This is inherent – if enough of the community wants to do something, the system allows it. So minorities must rely on social constraints and the reputation mechanism (majority members might lose rep if they abuse power). Still, it's a limitation that the protocol doesn't automatically protect against all harmful-but-popular decisions. It's possible for a community to effectively vote to dissolve itself or confiscate an

outsider's assets if that outsider is a small holder (some early DAO proposals in DeFi had aspects of this, though usually rational self-interest prevents too egregious moves).

6.5 Sybil Resistance of Reputation: The reputation system is a double-edged sword. It is supposed to reflect contributions, but measuring contributions automatically is hard. There's a risk of **gaming the reputation system**. Participants might try to artificially inflate their rep by spamming low-quality contributions that still earn points, or by forming a ring where they upvote each other if such mechanics exist. If reputation is accrued through on-chain actions, clever actors could create fake proposals or trivial votes to farm points ("participation farming"). The design can mitigate (like capping rep per period, manual review, etc.), but then manual steps reintroduce trust in whomever curates rep rewards. Also, reputation systems often have inherent biases and can unintentionally sideline newcomers (someone new might have great expertise but low initial rep, making it hard for them to have influence – it can ossify power with incumbents). Furthermore, reputation doesn't necessarily equate to competence across all issues: e.g. a high-rep developer might not be well-suited to decide on a marketing spend proposal, yet their vote still weighs heavily. TGC v2 doesn't currently differentiate domain-specific rep (though it could if extended as domain-specific delegation perhaps).

Another aspect: reputation can **decay** [18] to encourage continued activity, which is good, but if not calibrated it might punish those who take a break even if their expertise remains. Or if it doesn't decay, then over years a few early contributors hold high rep forever, again concentrating influence.

6.6 Technical and Security Risks: As with any smart contract system, there are risks of **bugs in the governance contracts** themselves. If the logic implementing TGC v2 has a flaw, it could be exploited (for instance, a bug in vote counting could allow someone to overflow their weight, or a bug in proposal execution could be hijacked to execute arbitrary code). The complexity of the system increases the attack surface. Thorough audits and formal verification would be needed to trust that the governance system is secure. If an exploit were found, ironically the remedy would be to use governance to patch it – but if governance itself is compromised, that's a catch-22. This scenario happened with the infamous "DAO hack" in 2016 – a bug in a DAO's code allowed funds theft, and because the governance couldn't quickly fix it (and was decentralized), it led to a chain fork. TGC v2's determinism means no quick manual override except if an emergency council mechanism is in place. We did include emergency proposals, which could be used to pause or fix the system if a critical bug is found, but that relies on the emergency council or similar to act swiftly. If such an emergency backdoor exists, it's also a trust point (who are those council members, and could they abuse their power under the guise of emergency?). We've assumed they are elected and limited in scope, but still it is a semi-centralization for dire cases.

6.7 Scalability: On-chain voting for every decision can be heavy. If the participant base is huge (millions of users), recording all their votes on-chain might be infeasible due to transaction throughput limits or cost. Some layer-2 or off-chain aggregation might be needed (e.g. vote commitments with Merkle trees to reduce load). Without such, TGC v2 might work best for communities up to a certain size unless the underlying chain is very scalable. There's also cost: each vote or proposal might require gas fees. If fees are high, that discourages participation (we've seen this on Ethereum – small token holders won't vote if it costs more in gas than their stake's value). Solutions include subsidizing votes or using a feeless chain, but those have trade-offs (spam, etc.).

6.8 On-Chain vs Off-Chain Gaps: Not everything can be solved on-chain. The governance process might need input from off-chain sources. For example, deciding on a partnership or legal contract – the chain can't enforce a real-world contract, it can only record an intention. So there's a gap when it comes to execution outside the blockchain: you still have to trust people to carry out certain decisions. TGC v2 in such cases can only make the decision transparent, but not enforce it. That said, our focus is on machine-executable governance, so that limitation is only for things you choose to govern that way.

Another subtle gap: **informal influence** – just because the formal structure is one-person-one-vote weighted by stake/rep, doesn't remove social influence outside the system. Charismatic or influential figures might sway votes regardless of their formal weight. If the founder of the project says "vote yes on this", many might follow blindly. TGC v2 doesn't prevent herd behavior or misinformation. Indeed, because proposals are open to discussion, misinformation campaigns could be waged (someone might misrepresent what a proposal does to get votes). The structure can record facts but not necessarily stop false narratives. Traditional governance has this issue too; it might be mitigated by requiring evidence or detailed proposals, but it's still a human factor limitation.

6.9 Legal and Regulatory Issues: As governance becomes code-based and global, it may clash with legal frameworks. For example, if TGC v2 is used in an organization incorporated in some jurisdiction, there might be requirements that contradict the on-chain method (e.g. law might require secret ballots for board elections, or might not recognize an on-chain vote as legally binding). If regulators demand the ability to intervene (say to freeze an account by law), a fully deterministic system doesn't have an easy hook for that (unless you build in a role for regulators, which then introduces trust in them). This might limit TGC v2 adoption in certain domains or require careful hybrid solutions.

In summary, TGC v2, while powerful, must be applied with an understanding of these limitations. Mitigating measures – some already mentioned – include: introducing optional secret voting for certain decisions (using cryptographic reveals to break the link for bribers), capping delegation accumulation or ensuring multiple independent delegates rather than one, continuously refining the reputation algorithm to resist gaming, providing better incentives or even requirements for participation (maybe slashing a small amount of stake for never voting, though that's contentious), and thorough security auditing. Also, ongoing monitoring is needed: the community should watch for centralization (if one delegate gets too powerful, others might proactively encourage redelegation to balance). Part of "trust is structure" is acknowledging when the structure might need reform – TGC v2 can always be amended by its own process if a certain limitation becomes critical (e.g. if bribery becomes rampant, the community could vote to implement secret ballots or quadratic voting as a change). Thus, TGC v2's greatest strength is that it can evolve to try addressing its shortcomings as they become apparent, provided the community remains vigilant and engaged.

Conclusion

We have presented **TGC v2: Machine-Layer Deterministic Trust Governance Protocol**, a comprehensive framework for structuring trust and governance in decentralized systems. TGC v2 builds on the insight that in trustless environments, mechanism design and cryptographic guarantees must replace traditional interpersonal trust. By encoding governance rules directly into the blockchain and ensuring every action follows those rules, TGC v2 provides an unprecedented level of transparency, accountability, and assurance in decision-making processes.

Key Contributions: This whitepaper detailed the full architecture of TGC v2. We defined a robust system model including participants with token stake and on-chain reputation, a dynamic delegation (trust) graph, and formally specified proposal state transitions. The protocol description in Section 3 outlined exactly how proposals are initiated, voted on with weighted votes (integrating both economic stake and meritocratic reputation), and executed automatically if approved – all under a consistent state machine that every node executes. We illustrated these processes with formulas for tallying and threshold conditions, demonstrating that governance decisions are deterministic functions of verifiable on-chain inputs. The emphasis throughout is on structural enforcement: **all changes are immutably recorded** on the ledger and **all participants can verify all states and events**, fulfilling the values of transparency and auditability [9] [8]. The catchphrase "**trust is structure**" was manifested in features like multi-signature

requirements (trust distributed among M-of-N parties) [2], liquid delegation (trust conferred via formal links rather than informal hierarchy) [15], and reputation-based weighting (trust earned through proven contributions) [20] [17]. Together, these mechanisms replace subjective or probabilistic trust models with deterministic, observable relationships.

Theoretical Significance: In Section 4, we analyzed why and how TGC v2 achieves strong guarantees. Notably, the protocol ensures that every honest node computing the governance state will arrive at the same outcome, eliminating ambiguity (which is crucial for a decentralized community to accept results as legitimate) [7]. We argued that TGC v2 can be seen as a confidence machine for governance, echoing academic perspectives that blockchain shifts trust to deterministic processes [1] [4]. We examined security aspects, concluding that no single actor or small coalition can subvert the system without detection or without meeting very high consensus thresholds – a stark contrast to off-chain governance where much depends on trusting committees or individuals. The combination of cryptographic immutability and multi-factor consensus creates high confidence that the system will "operate as planned, even if participants do not know or trust each other" [4]. We also highlighted the self-amendment capabilities: the protocol can govern and adjust itself, meaning it's not static but a living governance "constitution" subject to refinement by its community over time.

Practical Impact and Use Cases: Through Section 5's scenarios, we demonstrated TGC v2's versatility. From managing DeFi protocols and blockchain networks without human gatekeepers, to enabling new forms of collaborative governance in AI and consortia, TGC v2 opens avenues for more **trusted interactions in untrusted environments**. One overarching theme is that by providing a clear, enforceable governance framework, TGC v2 lowers the barrier for cooperation among parties who might not otherwise trust each other. They can trust the protocol and the structure it imposes. For example, a DAO using TGC v2 can assure its token holders that funds cannot be spent or rules changed except by the predefined supermajority vote – a level of assurance that is difficult to achieve even in traditional organizations (where embezzlement or unilateral decisions by insiders are always a risk). Another theme is democratization with accountability: liquid democracy via TGC v2 allows expertise to be leveraged (through delegation and rep) without giving up ultimate control by the masses, and it does so in a way that is transparent, countering some criticisms of pure representative systems. In short, TGC v2 can serve as the governance layer for **"crypto-civilization"** infrastructure [39] [40], aligning well with the vision of systems that are decentralized yet capable of complex, coordinated action.

Limitations and Future Work: We did not shy away from discussing challenges in Section 6. It is clear that TGC v2 is not a "set and forget" solution – like any governance framework, it requires an active, informed community to function optimally. Issues such as voter apathy, potential collusion, and the complexities of reputation systems will need ongoing attention. However, an advantage of TGC v2's deterministic approach is that problems become evident (e.g. low turnout is measurable, delegate concentration is visible) and thus can be addressed through further protocol tweaks or social efforts. Future work will explore enhancements like **privacy-preserving voting** (to mitigate bribery), more granular **domain-specific trust delegation** (so one can delegate technical matters to one person, financial matters to another), and **improved incentive schemes** to boost participation (perhaps via reward pools for voters or retroactive reputation for impactful decisions). Another important frontier is formal verification of the governance logic – proving properties like "no double-counting of votes, no unauthorized execution" in a mathematical sense, to bolster confidence. On the social side, experiments in different communities using TGC v2 will yield insights: we anticipate iterative improvements by observing real usage, much as blockchain consensus protocols evolved via practice.

Conclusion Statement: The innovations in TGC v2 push the boundary of what it means to have a "trustless" system. We often call blockchains trustless because they remove the need to trust transaction validators; TGC v2 extends that trustlessness to the realm of governance – removing the need to trust

benevolent leaders or committees, by entrusting governance to code and multi-party structures. In doing so, it provides a path toward more resilient and fair organizational structures, where rules cannot be secretly broken and power cannot be quietly usurped. All stakeholders, from the smallest to largest, share an immutable record and a voice in the process proportionate to their contribution and stake.

In essence, TGC v2 offers a blueprint for **"deterministic trust"**: the certainty that the system will behave according to its declared rules, and that any change in those rules will itself follow a rigorous, transparent process. We believe this model will be increasingly crucial as society entrusts more value and decision-making to decentralized digital platforms. By marrying the strengths of algorithmic decision-making with a rich conception of trust and reputation, TGC v2 aims to usher in an era where governance is both **machine-efficient** and **fundamentally trustworthy**.

We invite the community – researchers, developers, and potential adopters – to scrutinize, experiment with, and build upon TGC v2. Much like open-source software, open governance improves with collective input. Through collaboration, we can refine this protocol to better serve diverse communities and to truly realize the vision that **all aspects of a digital society can be governed without relying on blind trust – instead, trust will be built into the very structure of the systems we use**.

References

1. Algorand Foundation (2023). Governance dynamics: How on-chain and off-chain governance work. Algorand Blog. 9 8
2. De Filippi, P., Mannan, M., & Reijers, W. (2020). Blockchain as a Confidence Machine: The Problem of Trust & Challenges of Governance. Oxford Business Law Blog. 41 4
3. Nervos Network (2021). What Does Determinism Mean in Blockchain? Nervos Knowledge Base Article. 6 7
4. OSL (2025). What is Multi-signature? Multi-sig Wallets in Crypto Explained. OSL Academy. 2
5. Wikipedia (2023). Liquid democracy. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Liquid_democracy 15 12
6. Colony.io (2023). What is Reputation-Based Governance? Colony Blog. 20 17
7. TGC Whitepaper (2023). Governance & Community – Trust & Reputation. True Guard Coin Documentation. 16
8. TGC Whitepaper (2023). Decentralized Governance Structure (DAO Core Structure). True Guard Coin Documentation. 19
9. Tezos (2023). On-chain Stakeholder Governance. Gemini Cryptopedia (Author: A. B.). 14 23
10. Trust over IP Foundation (2022). ToIP Governance Architecture Specification V1.0. (Background on multi-layer trust frameworks). 36 37

---

1 3 4 34 41 Blockchain as a Confidence Machine: The Problem of Trust & Challenges of Governance | Oxford Law Blogs
https://blogs.law.ox.ac.uk/business-law-blog/blog/2020/10/blockchain-confidence-machine-problem-trust-challenges-governance

2 What is Multi-signature? Multi-sig Wallets in Crypto Explained
https://www.osl.com/hk-en/academy/article/what-is-multi-signature-multi-sig-wallets-in-crypto-explained

5 13 14 23 24 26 28 32 Tezos (XTZ): Superior Governance and Use Cases | Gemini
https://www.gemini.com/cryptopedia/what-is-tezos-xtz-governance-use-cases

6 7 33 What Does Determinism Mean in Blockchain?
https://www.nervos.org/knowledge-base/What_Does_Determinism_Mean_in_Blockchain_(explainCKBot)

8 9 25 Governance dynamics: How on-chain and off-chain governance work
https://algorand.co/blog/governance-dynamics-how-on-chain-and-off-chain-governance-work

10 11 Social Trusty Algorithm: A New Algorithm for Computing the Trust Score Between All Entities in Social Networks Based on Linear Algebra
https://www.mdpi.com/2076-3417/15/17/9744

12 15 31 Liquid democracy - Wikipedia
https://en.wikipedia.org/wiki/Liquid_democracy

16 21 Trust & Reputation | TGC white paper
https://tgc-4.gitbook.io/tgc-docs/governance-and-community/trust-and-reputation

17 18 20 22 What is Reputation-Based Voting in DAOs
https://blog.colony.io/what-is-reputation-based-governance/

19 29 Decentralized Governance Structure (DAO Core Structure) | TGC white paper
https://tgc-4.gitbook.io/tgc-docs/governance-and-community/decentralized-governance-structure-dao-core-structure

27 Multi-Layer Validator Network | TGC white paper
https://tgc-4.gitbook.io/tgc-docs/governance-and-community/multi-layer-validator-network

30 What is Liquid Democracy? A New Model for Agile Decision-Making ...
https://tokenminds.co/blog/knowledge-base/what-is-liquid-democracy

35 About TGC: Next-Gen Trusted Digital Infrastructure for the AI Era
https://tgc.plus/about.html

36 37 The ToIP Foundation Releases its First Official Governance Specifications - Trust Over IP
https://trustoverip.org/news/2022/02/01/the-toip-foundation-releases-its-first-official-governance-specifications/

38 Proposal: V2 of the Steward Council - DAO Governance and Vision
https://gov.gitcoin.co/t/proposal-v2-of-the-steward-council/11639

39 40 Foreword | TGC white paper
https://tgc-4.gitbook.io/tgc-docs/